2000

Classification Consequences of Preprocessing Radar Data

David J. Gorsich, Robert E. Karlsen, and Grant R. Gerhart
U.S. Army Tank-automotive and Armaments Command
AMSTA-TR-R, M.S. 263
Warren, MI 48397-5000

ABSTRACT

Support vector machines are classification algorithms based on quadratic programming that have been found to give excellent classification results on problems such as discriminating targets versus backgrounds. A key capability of these algorithms is that they need not require a preprocessing step to find feature vectors, yet preprocessing is still typically used. Preprocessing is still an important step in the classification process. We discuss what type of preprocessing steps are useful in improving the classifications results of support vector machines. We first give a short introduction to support vector machines. Then the algorithm is applied to the MSTAR radar data set. Several methods to preprocess the data are used before being sent to the support vector machine. The effect on the classification rate of the algorithm is then determined.

INTRODUCTION

The problem that is studied in this paper is radar image classification, and what types of preprocessing are useful before classification with a support vector machine begins. The support vector machine algorithm can take as input the entire digital image and classify it according to some criterion. As discussed in earlier papers, a pre-processing step is not necessarily necessary for feature extraction, but sometimes it can greatly improve the performance of the decision algorithm. Pre-processing is used to remove redundant information or to transform the image to a space where the objects are more easily classified.

Preprocessing or feature extraction can be performed using wavelets, peaks in the Fourier spectrum, filtering, preprocessing with conditioning matrices, statistical measures, histograms, etc. The number of methods that have been used is much longer than this. The question raised in this paper is, which pre-processing steps are actually useful. We consider the differences between linear and non-linear transforms of the feature vectors before classification begins. We find linear methods typically do not improve the classification performance. This includes all matrix operations and filtering operations on the training and test data set. We prove that linear operations cannot make non-separable data separable. On the other hand, non-linear methods can make large differences in the classification results, and can turn non-separable features vectors into separable ones.

The paper begins by providing a brief tutorial on Support Vector Machines (SVM's) and outlines derivations of some of the important elements. If already familiar with SVM's, the reader can skip directly to the section entitled Linear Preprocessing, which follows the SVM introduction. After the section on linear preprocessing, a short section on non-linear preprocessing follows. Then the SVM algorithm is applied to a publicly available Synthetic Aperture Radar (SAR) data set from the MSTAR program.

SUPPORT VECTOR MACHINES

Linearly separable data

The SVM algorithm[3-5] is based on finding the pair of parallel hyperplanes that separates the data into two classes while having the largest perpendicular distance between them. We conjecture that this will provide a good approximation to the "best" separating hyperplane, where by "best" we mean the hyperplane that will give, on average, the lowest classification error when new data is used.

To introduce the SVM's we first consider a linearly separable problem; i.e. the data can be separated completely by a hyperplane. Each data point is described by a feature vector, $\vec{x}$, and a truth value, y, that can take the values of +1 or −1, depending on the class. The two hyperplanes are required to pass through at least one point of each class and there can be no points between them. The boundary between the classes is then defined to be a third parallel hyperplane that is halfway between these two. The data points that the outer hyperplanes pass through, are called the support vectors, the meaning of which will be explained later. One of the hyperplanes consists of those points x which satisfy,

$$\vec{w}\cdot\vec{x}+b=+1, \tag{1a}$$

while the other hyperplane contains those points which obey,

$$\vec{w}\cdot\vec{x}+b=-1, \tag{1b}$$

with Eq. (1a) going through at least one point of class y=+1 and Eq. (1b) going through at least one point of class y=-1. The constants $\vec{w}$ and b define the hyperplanes, where $\vec{w}$ is a vector that is normal to the hyperplanes and $-b/\|\vec{w}\|$ is the perpendicular distance from the origin to the middle hyperplane. The RHS of Eq. (1a) will be greater than or equal to +1 for all points of class y=+1 and the RHS of Eq. (1b) will be less than or equal to -1 for all points of class y=-1. These can be combined into the following constraint on all the data points,

$$y_i\left(\vec{w}\cdot\vec{x}_i+b\right)-1\geq 0 \quad (1\leq i\leq n). \tag{2}$$

The perpendicular distance between the two outer hyperplanes is equal to $2/\|\vec{w}\|$. Therefore, finding the hyperplanes with the largest margin reduces to computing values for $\vec{w}$ and b that minimize $\|\vec{w}\|^2$, subject to the constraint in Eq. (2).

A standard procedure for handling optimization problems with constraints is given by the Lagrangian formalism[7-9]. The constraints are taken into account by adding multiples of the constraint equations to the objective function, which in this case, results in the following primal Lagrangian[3-5],

$$L_P=\frac{1}{2}\|\vec{w}\|^2-\sum_i\alpha_i y_i\left(\vec{w}\cdot\vec{x}_i+b\right)+\sum_i\alpha_i, \tag{3}$$

where α_i are the Lagrange multipliers associated with each of the constraints in Eq. (2). Because the constraints are inequalities, bounded from below, the Lagrange multipliers are required to be non-negative.[7-9]

Setting the derivative of the Lagrangian in Eq. (3) with respect to $\vec{w}$ and b (the primal variables) equal to zero, leads to the following expressions,

$$\vec{w}=\sum_i\alpha_i y_i\vec{x}_i, \tag{4a}$$

$$\sum_i\alpha_i y_i=0. \tag{4b}$$

Inserting Eqs. (4a) and (4b) into (3), results in the dual Lagrangian,

$$L_D=\sum_i\alpha_i-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j\vec{x}_i\cdot\vec{x}_j. \tag{5}$$

The problem is now reduced to finding the Lagrange multipliers (the dual variables) that maximize Eq. (5) and satisfy both the non-negativity constraints and the constraints of Eq. (4b). From the theory of constrained optimization ones finds that the only constraints that matter are those that actually constrain the minimization of the objective function. The other (inactive) constraints can be discarded without changing the optimal point. It is consistent therefore to set the Lagrange multipliers for the inactive constraints to zero[8,9]. This condition can be summarized as:

$$\alpha_i\left(y_i\left(\vec{w}\cdot\vec{x}_i+b\right)-1\right)=0, \tag{6}$$

which means that those data points with non-zero Lagrange multipliers (and hence are active constraints) will lie on the outer hyperplanes. These data points are called the support vectors and they are the points that determine the position of the hyperplanes. One can move the other points around the feature space or remove them entirely and the solution will not change, provided one does not move a point across one of the outer hyperplanes.

One can solve Eq. (5) using any quadratic programming solver, although different solvers perform better on different types of problems[5,7-9]. Solving the quadratic programming (QP) problem efficiently is actually one of the most difficult parts of SVM and there exists many numerical QP solvers that are readily available.

Once the Lagrange multipliers are known, the solution for $\vec{w}$ is given by Eq. (4a), where the sum can be restricted to the support vectors, since they are the only ones with non-zero α. One can find b from Eq. (6) using any of the support vectors, although one generally averages over all the support vectors for better accuracy. Once these constants are known, the classification of an unknown vector, $\vec{v}$, is given by the sign of,

$$b+\sum_i \alpha_i y_i \vec{x}_i\cdot\vec{v} \quad (1\le i\le n_{sv}), \tag{7}$$

where the sum is over the support vectors.

Nonlinearly separable data

Now suppose that the "best" boundary between the data is nonlinear. An example of this situation is shown in Fig. 1, using a non-homogeneous quadratic kernel. One cannot separate the two classes with a straight line. The structure of the SVM equations allows a simple solution to this situation. Map the data, through a nonlinear transformation ϕ to a different space where the data can be separated with a hyperplane. This results in the Lagrangian in Eq. (5) being transformed to[3-5],

$$L_D=\sum_i \alpha_i-\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \phi(\vec{x}_i)\cdot\phi(\vec{x}_j). \tag{8}$$

Since Eq. (8) depends only on scalar products between the transformed feature vectors, one can replace the scalar product with a kernel function,

$$K(\vec{x},\vec{y})=\phi(\vec{x})\cdot\phi(\vec{y}), \tag{9}$$

and never need to compute the transformation ϕ explicitly. Equation (7) then becomes,

$$b+\sum_i \alpha_i y_i K(\vec{x}_i,\vec{v}) \quad (1\le i\le n_{sv}), \tag{10}$$

with the test feature vector now inside the summation over the support vectors.

$$K(\vec{x},\vec{y})=(\vec{x}\cdot\vec{y}+1)^2. \tag{11}$$

Non-separable data

A typical problem in applications is that the data is not separable using a given kernel. Sometimes, no moderately smooth kernel can separate the data. Then the assumptions leading to Eq. (1) no longer hold. Although the preceding SVM algorithm can provide a reasonable solution in these cases, many times the separating hyperplane is not one that would be considered the "best" solution. This is due to the outliers being given more weight than the other data points. To avoid this, positive

slack variables δ are introduced that measure the deviation of the outliers from the optimal separating hyperplanes. The constraint equation (2) then becomes,[3-5]

$$y_i\left(\vec{w}\cdot\vec{x}_i+b\right)-1+\delta_i\geq 0. \qquad (12)$$

A convenient way to minimize the total amount of the outlier error is to add an appropriate term to the Lagrangian, which is multiplied by a weighting factor C. Choosing the error term to be the sum of the deviations leads to the optimization solution being independent of the slack variables and their associated Lagrange multipliers. The only effect of this additional term is to restrict the original Lagrange multipliers to, $0\leq\alpha_i\leq C$, instead of being simply non-negative. Lower values for C correspond to smaller penalties for outliers and a softer margin. The threshold b can still be found from Eq. (6), provided the corresponding Lagrange multiplier is not at the upper bound C.

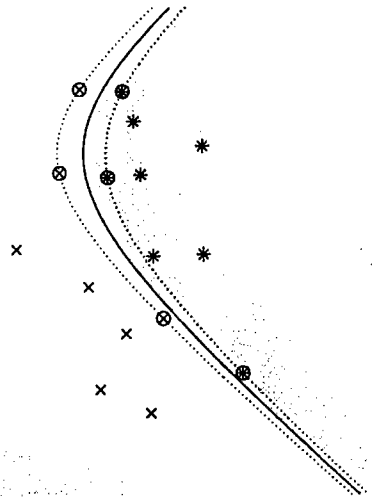

Figure 1: Non-linear separable data

LINEAR PREPROCESSING

We start the discussion of preprocessing by considering linear preprocessing operations on non-linearly separable data. Linear operations are essentially any matrix operation of the feature vectors before the SVM operation,

$$X' = XR, \qquad (13)$$

where R is a matrix acting on the feature vectors that lie on each row of the feature vector matrix X, with $\vec{x}_i$ in each row. The matrix R can be a filtering matrix, or any other operation that is linear. For example, take a totally random matrix R and apply it to the feature vectors in Figure 1. The entries of the matrix R are taken as Gaussian random variables. Since X is a 16x2 matrix, R is a 2x2 matrix. There are only two columns because we are looking at an example of feature vectors in two dimensions. Since a random 2x2 matrix R will rarely have a zero eigenvalue, it is invertible. The key question is, what is the effect on the SVM of using the preprocessed feature vectors X' given by Eq. (13)? The answer can partially be found using linear algebra. In one realization of a random matrix R,

$$R=\begin{vmatrix} .4387 & .2140 \\ .4983 & .6435 \end{vmatrix}, \qquad (14)$$

the resulting SVM classification using the modified feature vectors of Figure 1 are displayed in Figure 2. The classification of the feature vectors is again perfect. The classes are correctly identified, the number of support vectors are the same and the feature vectors are still non-linearly separable. Is this true in general? That is, for any feature vector matrix X and any matrix R? The answer is almost, yes. The answer is found by considering what the action of any matrix on any other matrix is, and by using the singular value decomposition. The singular value decomposition theorem states any matrix can be broken into two orthogonal matrices, and a diagonal matrix of singular values in the middle,

$$R = USV. \qquad (15)$$

This means the action of any matrix R is to rotate the feature

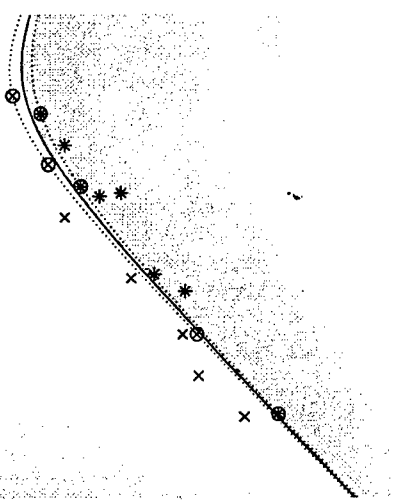

Figure 2: Feature Vectors that have been pre-processed with a linear operator are still non-linearly separable.

vectors X, using U. Then to stretch the basis or axes by a factor given by the singular values S, and finally to again rotate what is left by V. The result can be seen in Figure 2 for the matrix R given in Eq. (14). Generally, linear preprocessing will not help improve the classification algorithm. For example, we could not expect a matrix R to turn the problem in Figure 1 into a linearly separable one, since R only rotates and stretches the axes. It is not possible to find a matrix R to turn a linearly non-separable problem into a linearly separable one. Of course using a singular matrix R, we could turn a linearly separable one into a non-separable one, but this would degrade the performance of the SVM.

Earlier we mentioned the answer would only be partially found. We concluded that a linear operation R on the feature vectors X would not improve the classification rate, largely because it does not change the problem theoretically. But the matrix R can change the numerical aspects of the problem. In solving the SVM problem, the Hessian matrix often has a large condition number, so the matrix must be regularized. Numerically, the number of support vectors can change when the matrix X is preprocessed with a matrix R. This is most clearly seen when the eigenvalues or singular values of R have a great disparity in magnitude. This means in the two dimensional case, one axis is squished, and the other is greatly expanded, pushing all the feature vectors into a line, very close together. Then regularizing, and other numerical issues change the solution. Unfortunately, this generally worsens the solution unless it is known that the feature vectors are squished into a subspace. If that subspace can be found, expanding along that direction may help the numerical aspects of the solution. These ideas will be left for a latter experiment. It may be found that if a linear operator improves the classification rate, it is in fact just improving the numerical performance of SVM, and not increasing the information contained in the feature vectors.

NON-LINEAR PREPROCESSING

Most preprocessing steps fall into the non-linear category including wavelets and multiresolution analysis because those algorithms subsample the data. Other non-linear methods are various statistical measures, histograms, the median filter and morphological operations. Of course these methods can all have a large impact on the classification rate, and greatly change the amount of information in the feature vectors. As an example, consider Figure 3. The feature vectors from Figure 1 are transformed by a spreading function near the center of the one class with most of its points on the right. This function is performed by adding an exponential with a sign factor in front so that points near the off-center thus pushing them further away than points farther from the center of the one class. It is generally difficult to find the right preprocessing type, but it is certainly possible to find a preprocessing step that creates linearly separable data from non-linearly separable data. An automatic method of determining the non-linear preprocessing step is a future research direction.

MSTAR RADAR DATA RESULTS

We start by training on a subset of the MSTAR data set, and assuming there are only two classes, T72s and non-T72 radar signatures, the BMP. By using azimuth index data, we only use radar signatures between 0 and 15 degrees. This means typically between 6 and 9 images per vehicle. A total of 47 full sized radar images were used to train the data, and another 46 different images to test the data.

In each case, 22 images were BMP. In the training set there were 25 T72 images, and there were 24 in the test set at those angles. The test set does not use the same BMP and T72 vehicles. The radar images are normalized and the mean is taken out, as was done in recent papers by Karlsen[13,16]. Training and testing was done with a degree 2 polynomial. The training set was completely classified correctly. The testing data set was classified correctly 67.42% of the time, 68.18% classification of the BMPs and 66.67% classification of the T72s. The goal was not to reach a high classification rate, but to determine the effects of linear and non-linear operators used for preprocessing. So far, no real preprocessing was done on the images. The raw images are being used by the classifier (other than normalization). What

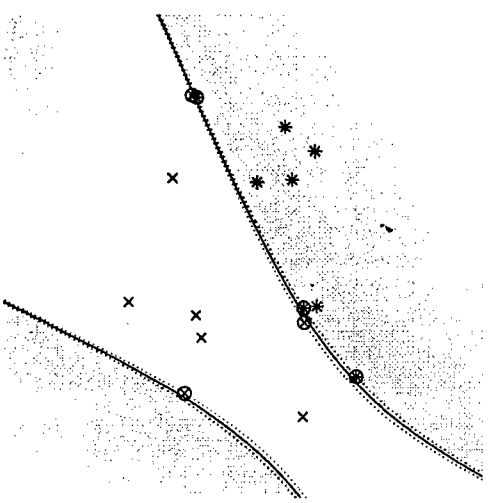

Figure 3: Feature Vectors that have been pre-processed with a non-linear stretching operator and are nearly linearly separable.

happens to the classification rate as random Gaussian matrices (invertible) are applied to the data? This was done ten times to the training and testing feature vectors, and classification rates were re-evaluated. For training, the feature matrix X was 47x1024 (each row is a radar image), so operating on columns means we take R to be 1024x1024. After using a symmetric random matrix R on both training and testing over 10 simulations. The training data set was again always correctly classified. On the testing data set, the average classification rate was 61% with a standard deviation of 2.5%. The mean classification rate of BMPs was 77.26% and the T72, 44.16% with standard deviations of 5.5% and 4.7%. This means that the matrix R is effecting the classification rate, but not by a great deal as expected. A non-linear preprocessing step can of course greatly change the classification results.

CONCLUSIONS

In this paper, we introduced the notion of support vector machines, and as an example, classified a subset of the MSTAR data base. Support vector machines typically use a pre-processing step on the feature vectors before classification begins, but do not need to do this. SVMs can directly train and test on the radar images themselves. We considered the consequences of preprocessing on the classification rate of SVMs. We determine that linear operations have little effect of the classification result theoretically, but numerically can make a difference. Of course, non-linear operations can make a big difference on classification rates. We test our results on a small subset of the MSTAR database and find classification rates to change by only a standard deviation of 2.5%.

REFERENCES

[1] R.E. Karlsen et. al, "Comparative study of wavelet methods for ground vehicle signature analysis," *Wavelet Applications III*, ed. H.H. Szu, SPIE Proc. **2762**, 314-24 (1996).

[2] D.J. Gorsich, R.E. Karlsen, G.R. Gerhart and M.G. Genton, "Target versus background characterization: Second generation wavelets and support vector machines," *Targets and Backgrounds: Characterization and Representation V*, eds. W.R. Watkins, D. Clement and W.R. Reynolds, SPIE Proc. **3699**, 185-96 (1999).

[3] V. Vapnik, *Statistical Learning Theory*, John Wiley and Sons, New York (1998).

[4] C.J.C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery* **2**, 1-47 (1998).

[5] B. Schölkopf, C.J.C. Burges, and A.J. Smola, eds., *Advances in Kernel Methods, Support Vector Learning*, MIT Press, Cambridge MA (1999).

[6] Y. Lecun et al., "Comparison of learning algorithms for handwritten digit recognition," *Proc. Int. Conf. Artificial Neural Networks* **II**, eds. F. Fogelman-Soulié and P. Gallinari, 53-60 (1995).

[7] D.G. Luenberger, *Linear and Nonlinear Programming*, Addison-Wesley, Reading MA (1984).

[8] R. Fletcher, *Practical Methods of Optimization*, John Wiley and Sons, Chichester (1987).

[9] D.P. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont MA (1995).

[10] V. Vapnik, *Estimation of Dependences Based on Empirical Data*, Springer-Verlarg (1982).

[11] E. Osuna, R. Freund and F. Girosi, "An improved training algorithm for support vector machines," *Proc. IEEE Neural Networks for Signal Processing* **VII**, eds. J. Principe et. al, 276-85 (1997).

[12] J.C. Platt, "Sequential Minimal Optimization: A fast algorithm for training support vector machines," Tech Report MSR-TR-98-14, Microsoft Research (1998).

[13] R.E. Karlsen, D.J. Gorsich and G.R. Gerhart "Target classification and support vector machines", Proc. 10th Annual Ground Target Modeling and Validation Conf., Houghton MI, 286-94 (1999).

[14] H. Demuth and M. Beale, *Neural Network Toolbox User's Guide*, The Mathworks Inc., Natick MA (1998).

[15] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," *Proc. IEEE Int. Conf. Neural Networks*, San Francisco (1993).

[16] R.E. Karlsen, D.J. Gorsich and G.R. Gerhart, "Target classification via support vector machines," *Optical Engineering* **39**(3), 704-11 (2000).

[17] See http://www.mbvlab.wpafb.af.mil/public/sdms/.

[18] V. Velten, T. Ross, J. Mossing, S. Worrell and M. Bryant, "Standard SAR ATR Evaluation Experiments using the MSTAR Public Release Data Set", *Algorithms for Synthetic Aperture Radar Imagery V*, SPIE Proc. **3370**, (1998). (Available at http://www.mbvlab.wpafb.af.mil/papers/SSAEEutMPRDS/mstar_public_eval.html)

[19] C.J.C. Burges, "Simplified support vector decision rules," *Proc. 13th Intl. Conf. on Machine Learning*, ed. L. Saitta, 71-77 (1996).

16255

OPSEC REVIEW CERTIFICATION

(AR 530-1, Operations Security)

I am aware that there is foreign intelligence interest in open source publications. I have sufficient technical expertise in the subject matter of this paper to make a determination that the net benefit of this public release outweighs any potential damage.

Reviewer: Thomas Meitzler | 14 | Research Scientist

Name | Grade | Title

Thomas Meitzler | 8/3/00

Signature | Date

Description of Information Reviewed:

Title: CLASSIFICATION CONSEQUENCES OF PREPROCESSING RADAR DATA

Author/Originator(s): DAVID GORSICH, ROBERT KARLSEN, GRANT GERHART

Publication/Presentation/Release Date: 16AUG00

Purpose of Release: CONFERENCE PRESENTATION & PROCEEDINGS (GTMV)

An abstract, summary, or copy of the information reviewed is available for review.

Reviewer's Determination (check one)

(1) Unclassified Unlimited.

2. Unclassified Limited, Dissemination Restrictions IAW _____

3. Classified. Cannot be released, and requires classification and control at the level of _____

Security Office (AMSTA-CM-XS):

Concur/Nonconcur _____ Signature | 11 AUG 00 Date

Public Affairs Office (AMSTA-CM-PI):

Concur/Nonconcur _____ Signature | 21 AUG 00 Date

STA 7114-E
Sep 1999